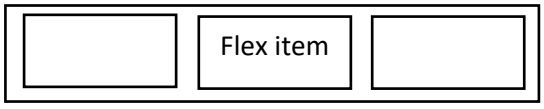
Flexbox

The flexible box module is usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between item in an interface and powerful alignment capabilities.

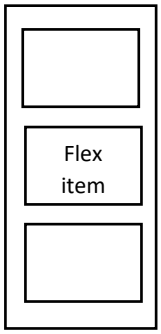
Flex Dimension

1. Row



Flex container

2. Column



Flex container

Flexbox Properties

1. Property For the Parent (Flex container)
   1. display

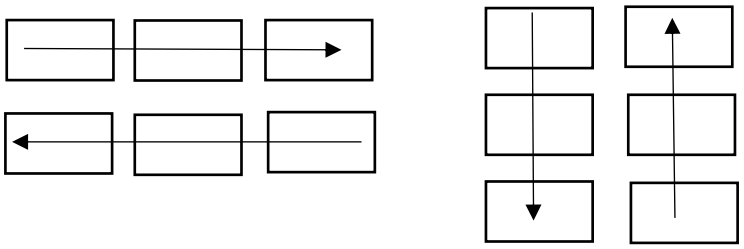
This define a flex container inline or block depending on the given value. It enables a flex context for all its direct children.

```
.container
{
        display:flex;
}
```

/* or inline-flex */

❖ Note
Css column have no effect on a flex container.

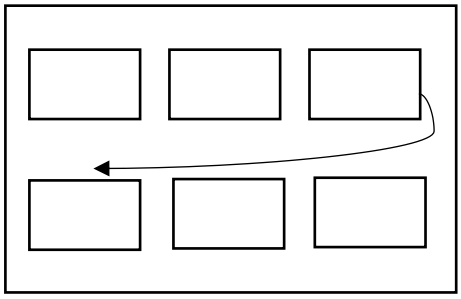
2. flex-direction



Flex box is a single direction layout concept. Flex items as primarily laying out either in horizontal rows or vertical columns.

```
.container
{
flex-direction: row/ row-reverse/column/ column-reverse;
}
```

3. flex-wrap



By default, flex item will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

```
.container
{
flex-wrap : nowrap/wrap/wrap-reverse;
}
```

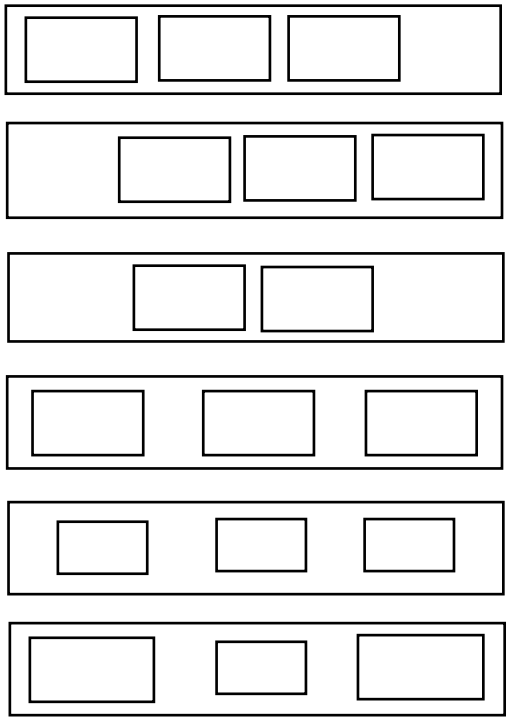
4. flex-flow

   This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. The default value is rownowrap.

```
.container

{

flex-flow : column wrap;

}
```

5. Justify-content

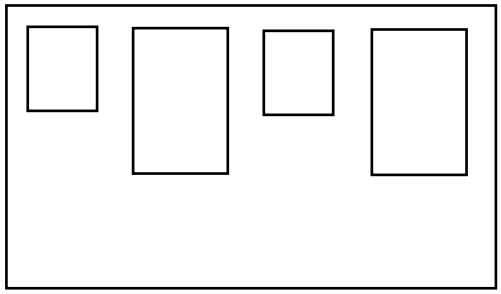
   - ❖ flex-start
   - ❖ flex-end
   - ❖ center
   - ❖ space-between
   - ❖ space-around
   - ❖ space-evenly



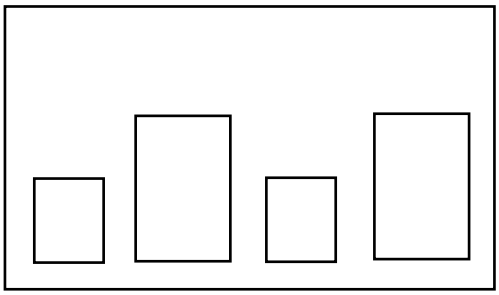
```
.container

{

Justify-content : flex-start;

}
```

6. align-items

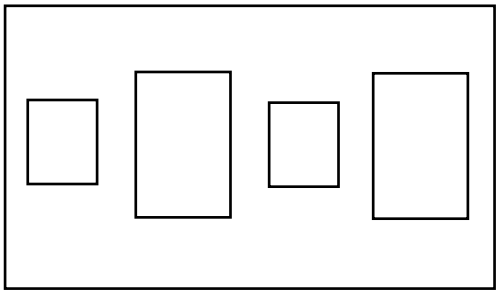
*flex-start



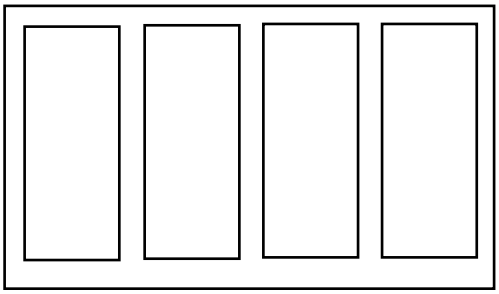
* flex-end



*center



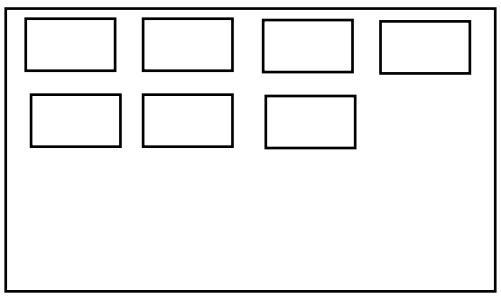
* stretch



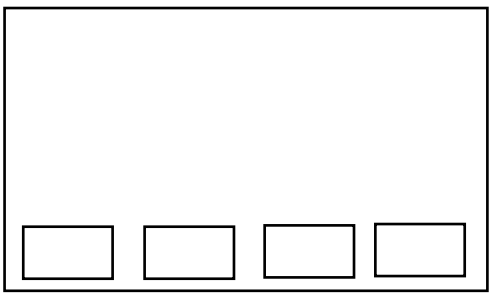
```
.container
{
align-items: stretch;
}
```

7. align-content

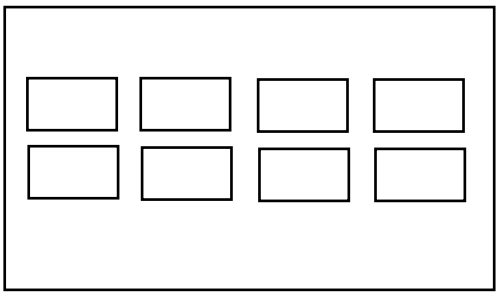
*flex-start



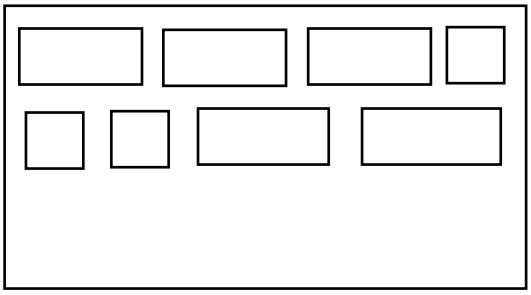
* flex-end



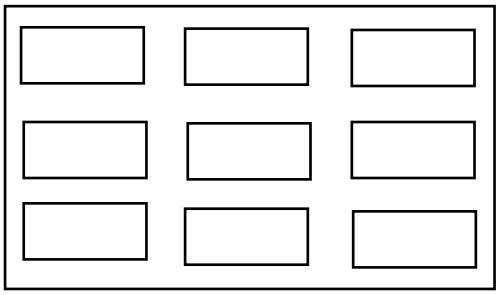
*center



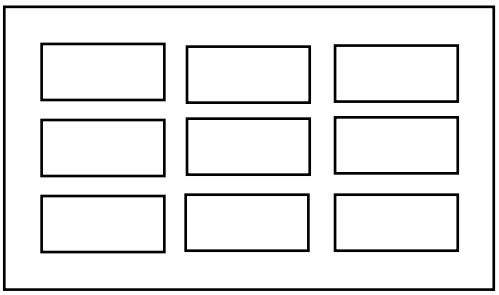
* stretch



*space-between

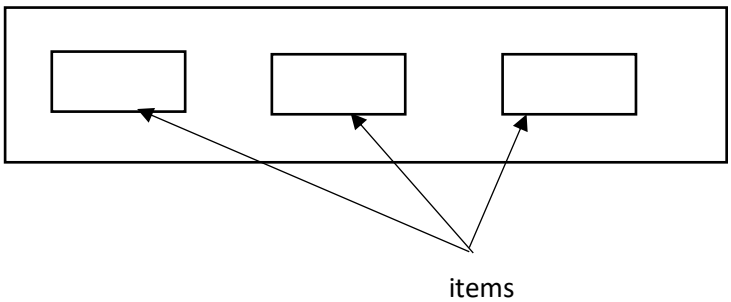


*space-around

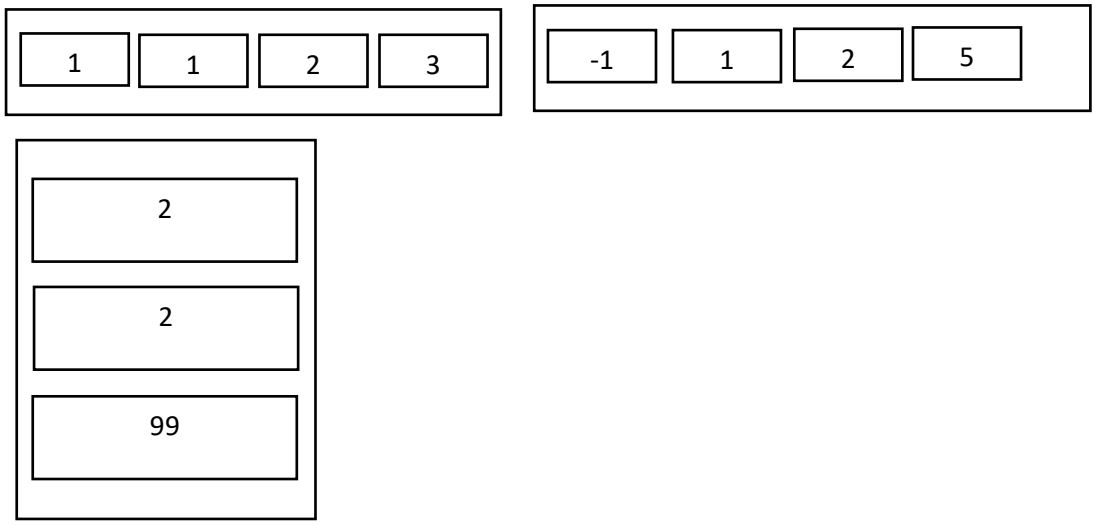


```
.container

{

align-content: flex-start;

}
```

2. Properties for the children (flex items)

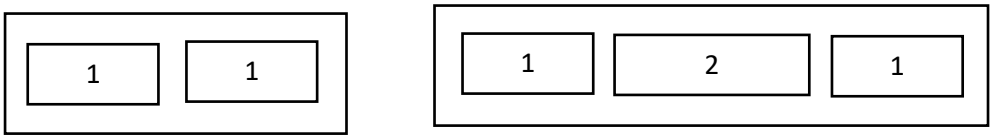


1. Order




```
.item
{
order: 5; /* default is 0 */
}
```

2. flex-grow




```
.item
{
flex-grow: 4; /* default is 0 */
}
```

- Note : -ve numbers are invalid.

3. flex-shrink

This defines the ability for a flex item into shrink if necessary.

```
.item
{
flex-shrink: 3;
}
```

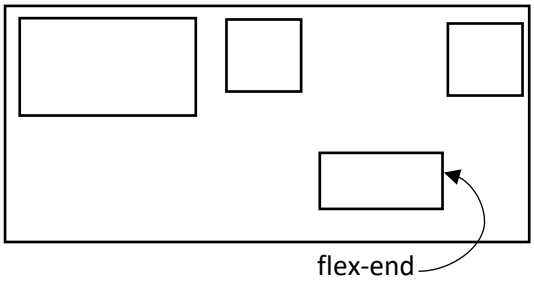
4. flex

This is short hand for flex-grow, flex-shrink and flex-basis combined.

```
.item
{
flex: none| [<'flex-grow'><'flex-shrink'>?||<'flex-basis'>]
}
```

5. flex-start

flex-start




```
.item
{
align-self: auto| flex-start| flex-end| center | baseline| stretch;
}
```

- Note: float, clear and vertical-align have no effect on a flex item.

❖ Example 1:

```
.parent
{
   display: flex;
   height:  300px;
}
.child
{
    width: 100px;
    height:100px;
    margin: auto;
}
```

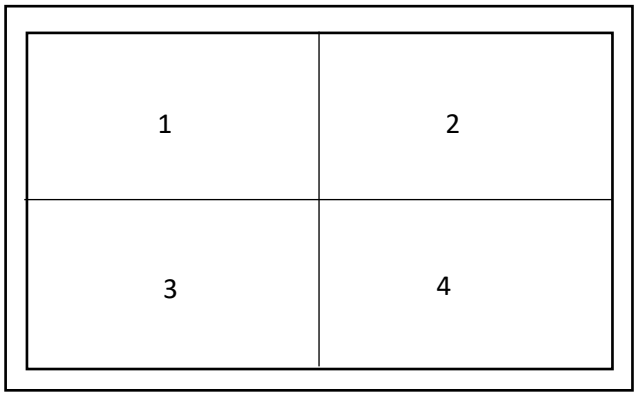
❖ Example 2

```
.navigation
{
   display: flex;
   flex-flow: row wrap;
   justify-content: flex-end;
}
@media all and (max-width: 800px)
{
.navigation
{
    Justify-content: space-around;
}
}
@media all and (max-width: 500px)
{
.navigation{
   flex-direction: column;
}
}
```

CSS Grid Layout Model

This layout model offers a grid based layout system, with rows & columns, making it easier to design webpage without having to use floats & positioning.

Example:



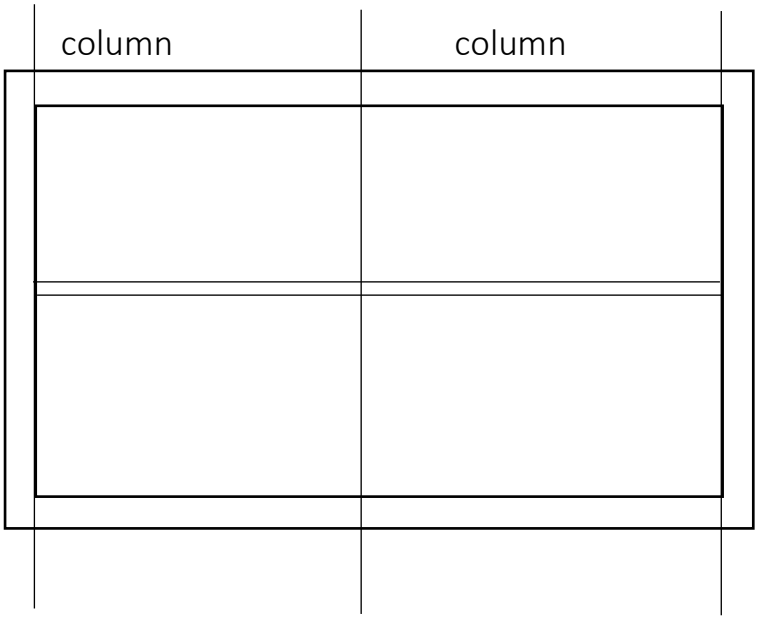

```
<style>
.grid-container
{
  display:grid;
  grid-template-columns: auto auto;
  background-color: blue;
  padding: 10px;
}
.grid-item
{
  background-color: pink;
  border 1px solid white;
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
<div class=”grid-container”>
<div class=”grid-item”>1</div>
<div class=”grid-item”>2</div>
…..
</div>
```

Display Property

An Html element becomes a grid container when its display property is set to grid or inline-grid.

Grid Columns

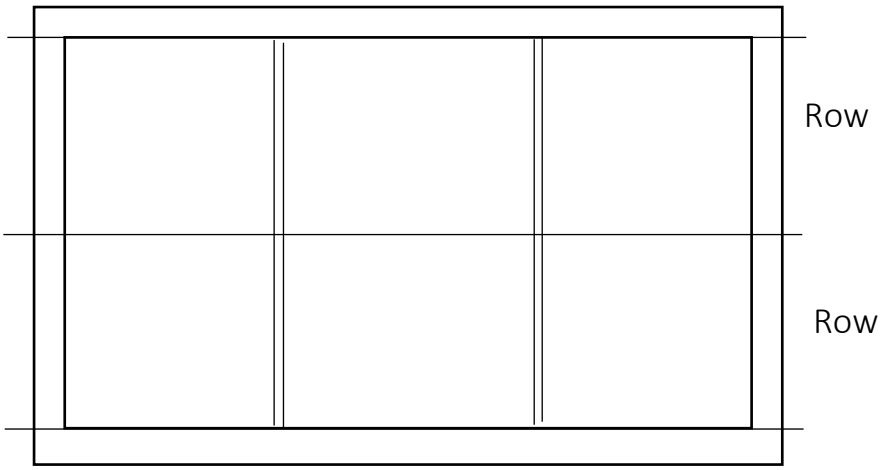
The vertical lines of grid items are called columns.




```
.container
{
column-count: 3;
}
```

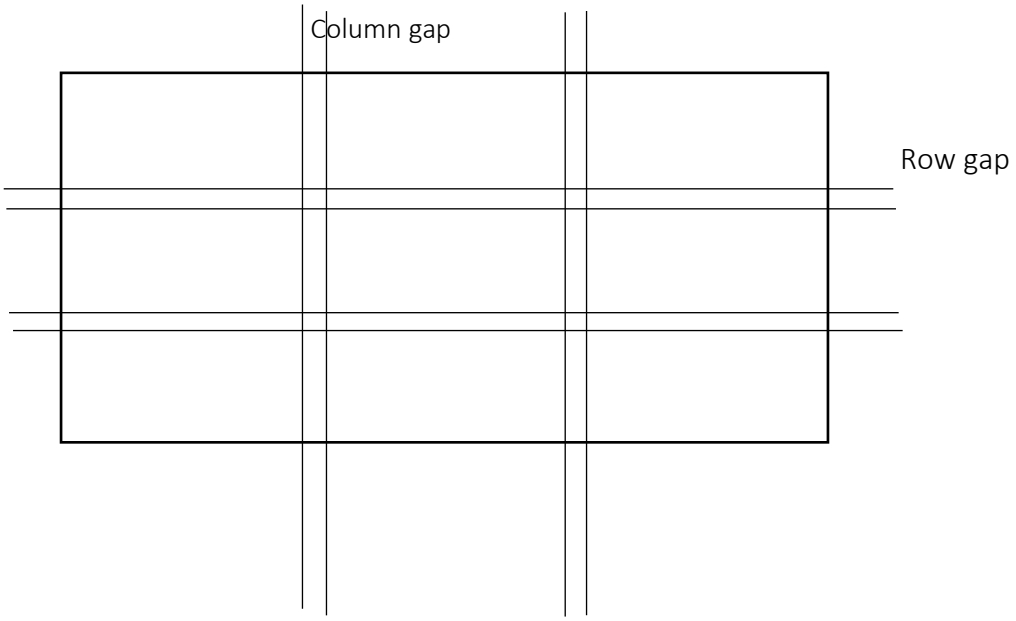
Grid Rows

The horizontal lines of grid items are called rows.



Grid Gap

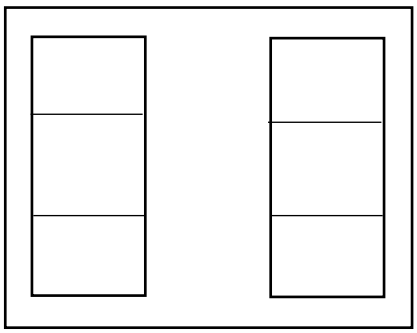
The space between each column / row called gaps.



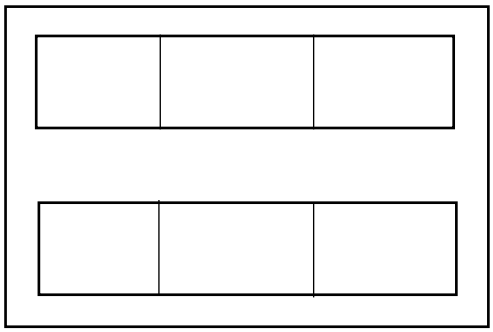
Adjust the gap size by using one of the following properties:

1. grid-column-gap
2. grid-row-gap
3. grid-gap

Example: grid-column-gap:50px

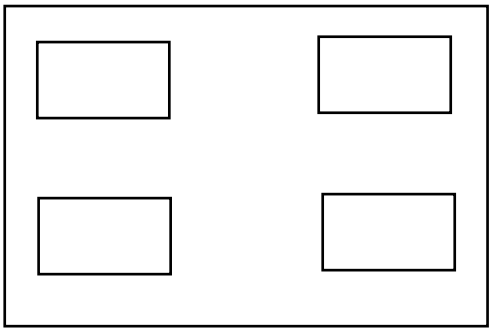


Example: grid-row-gap:50px



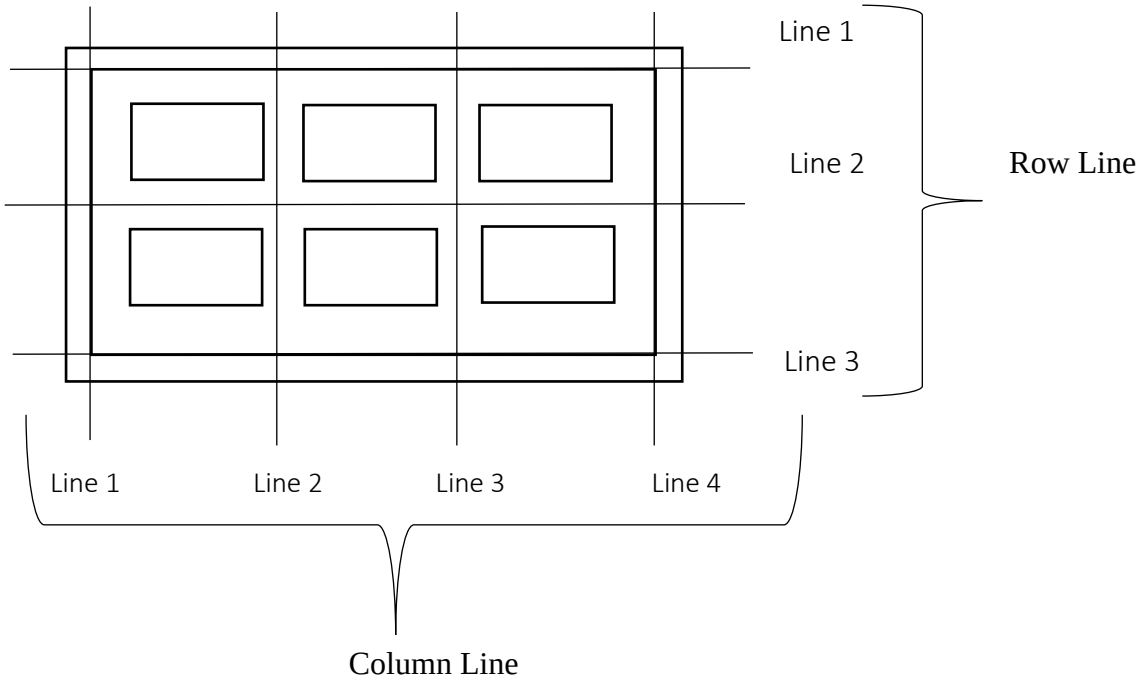
Example: grid-gap:50px 100px;

[short hand property to adjust a space between the rows & columns]



Grid Line

The line between row are called row line & column are called column line.



Example:

1		2
3	4	5
6	7	8

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
</style>
</head>
<body>
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
```

</body>

Css Grid Properties

1. column-gap : specific gap between the column.
2. gap : short hand property for the row gap & column gap.
3. grid : a short hand property for the grid-template-rows, grid-template-columns, grid-auto-flow.
   Example: grid: 10px| auto;
4. grid-area :
   Example: .item1{
   grid-area: 2 / 1 / span2 / span3;
   }
   [item will start with row 2 & column 1 & span 2 rows and span 3 columns]

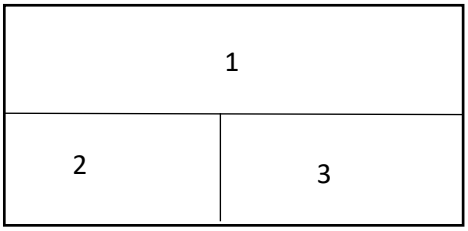
Definition & usage

grid-area property specifies a grid item`s size and location in a grid layout, and it shorthand property as following properties:

- grid-row-start
- grid-column-start
- grid-row-end
- grid-column-end

We can use grid-area property to name grid item.

Example:
```
.item1 {
 grid-area: myArea;
}
.container {
 grid-template-area: ‘myArea myArea’;
}
```

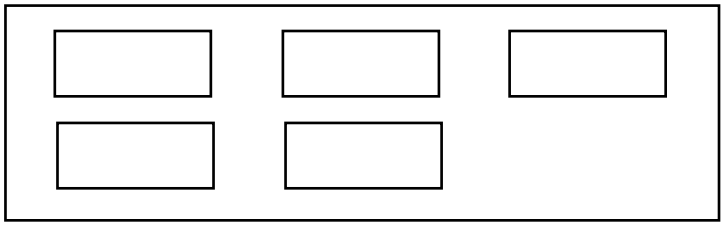


5. grid-auto-flow

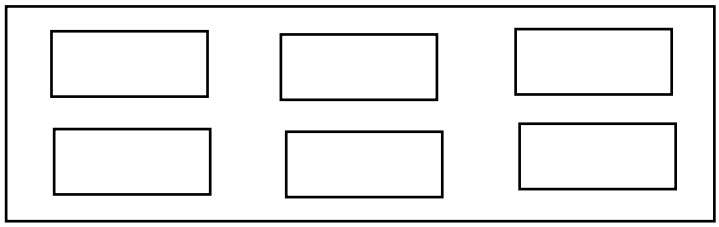
Example: .grid-container

```
{
    display:grid;
    grid-template-column: auto auto auto;
    grid-template-row: auto auto auto;
}
```

```
<div class=”grid-container” style=”grid-auto-flow:row;”>
:
:
</div>
```



```
<div class=”grid-container” style=”grid-auto-flow:column;”>
:
:
</div>
```



6. grid-column property

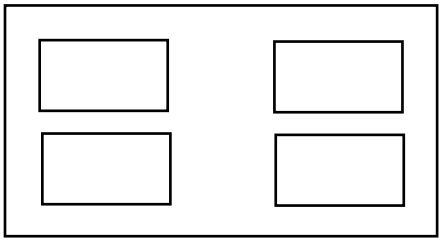
Example: .item1

```
{
    grid-column: 1/ span 2;
}
```

1		2
3	4	5

7. grid-column-gap

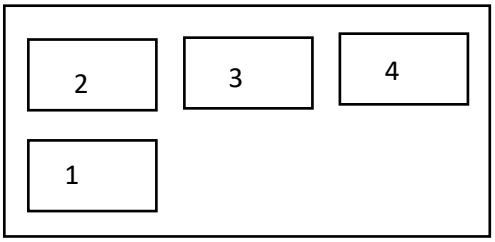
   Grid column gap property to specify the size of the gap between the column.



   Example: .container

```
{
    display:grid;
    grid-template-column: auto auto;
    grid-column-gap: 50px;
}
```

8. grid-row-start



   Example: .item1

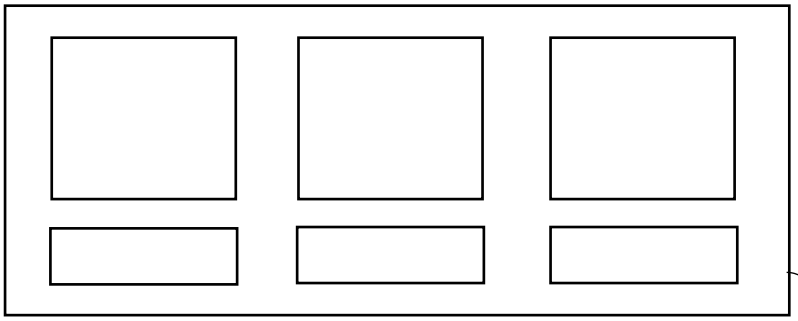
```
{
    grid-row-start: 2;
}
```

9. grid-template

   It is a short hand property for the grid template rows and grid template column.

   Example: .container

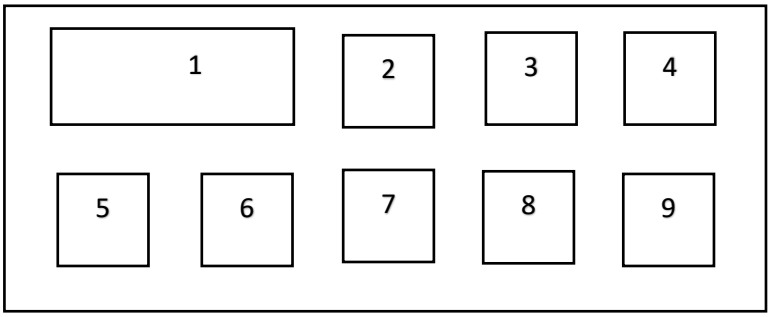
```
{
    display:grid;
    grid-template:150px / auto auto auto;
}
```



Grid layout has three columns and the first rows has 150px height.

10. grid-template-area

Example: Item1 is called myItem and it will take up the place of 2 columns out of 5.



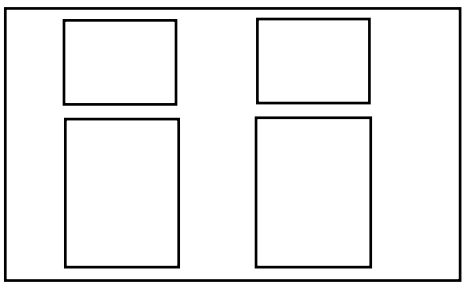

```
.Item1
  {
    grid-area: myarea;
  }
.gridcontainer
  {
    display:grid;
    grid-template-area:` myarea   myarea…….`;
  }
```

11.grid-template-column

Number of column in grid layout.
Example: grid-template-column: auto auto;

12. grid-template-row

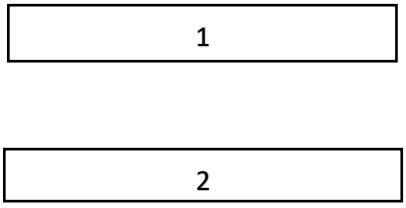
Example:
```
.grid-container
    {
      display:grid;
      grid-template-columns: auto auto;
```



```
grid-template-rows: 100px 300px;
}
```

13. row-gap

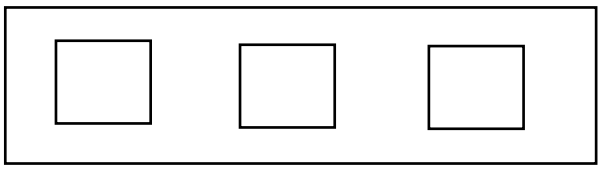
Example:
```
.gridcontainer
{
    display:grid;
    row-gap:50px;

}
```



CSS Grid Container

1. display - grid / inline grid
2. grid-template-column
3. grid-template-row
4. justify-content
   Alignment of the grid inside the container.

Example: justify-content:space-evenly;



```
justify-content: space-around;
justify-content: space-between;
justify-content: center;
justify-content: start;
justify-content: end;
```

The align-content property

Example: align-content: center;

Note

The grids total height must be less than the containers height for the align-content property to have any effect.

Example: .grid-container

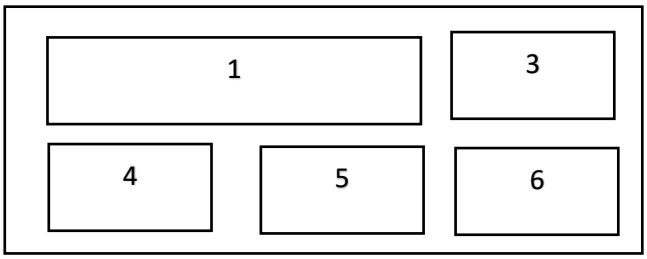
```
{
    display: grid;
    height : 400px;
    align-content: center / space-evenly / space-around /
                   space-between / start / end ;
}
```

CSS Grid Item

Child elements (items)

A grid container contains grid items.

1. The grid-column property



Example: .Item1

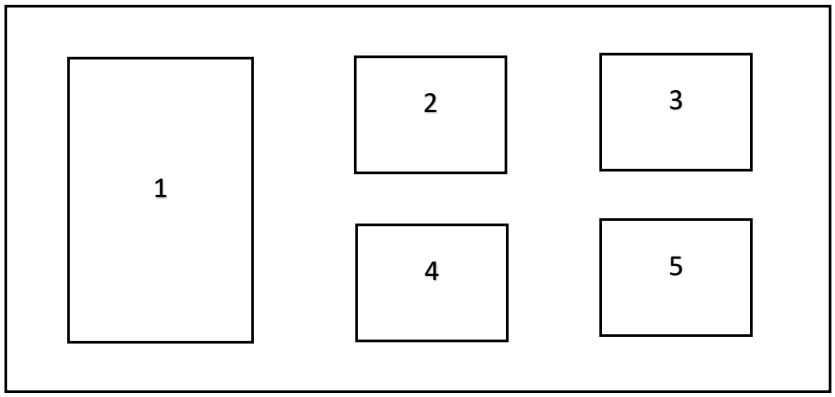
```
{
    grid-column: 1 / 3; [ Item will start at column1 &
                          end before column 3]
}
```

.Item1

```
{
    grid-column: 1 / span 2;
}
```

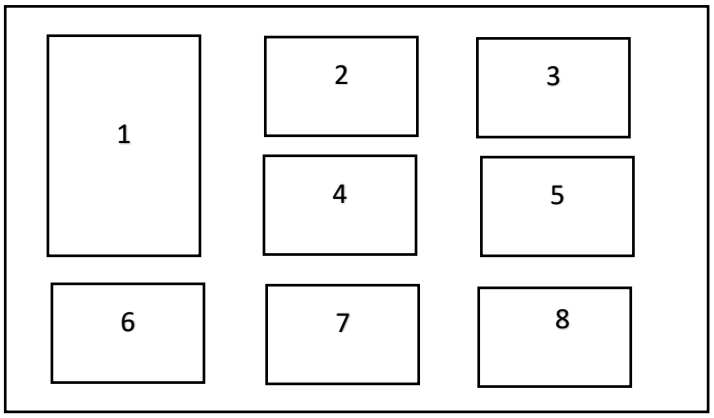
2. The grid-row property

Example:



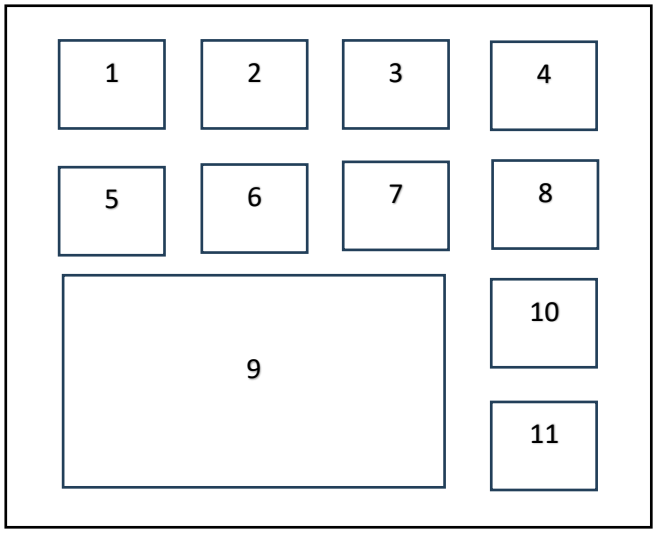

```
.Item1
  {
    grid-row: 1 /3; [ Item1 start on row-line 1 & end on row-line 4]
   }
```

```
.Item1
    {
      grid-row: 1/ span 2;
    }
```



3. The grid-area property

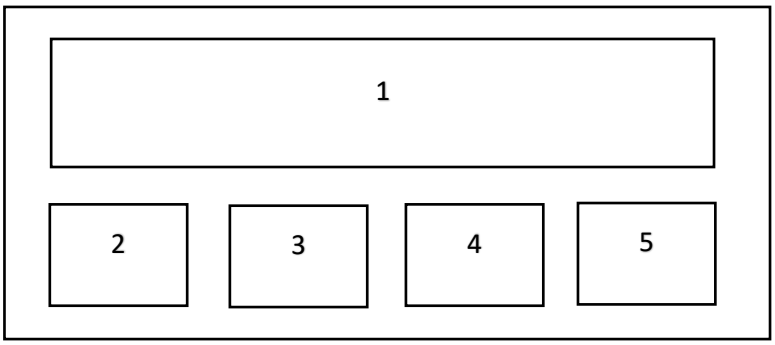
```
.Item8
   {
    grid-area: 1 / 2 / 5 / 6; [grid-row-start / grid-column-start /
                               grid-row-end / grid-column-end]
    }
.Item9
    {
     grid-area: 3 / 1 / span 2 / span 3;
     }
```



Naming Grid Items

Example: .Item1

```
{
  grid-area: myarea;
}
.gridcontainer
{
   display: grid;
   grid-template-areas: ‘myarea myarea myarea myarea’;
}
```



Note 1

Each row is defined by apostrophes (` `).

Note 2

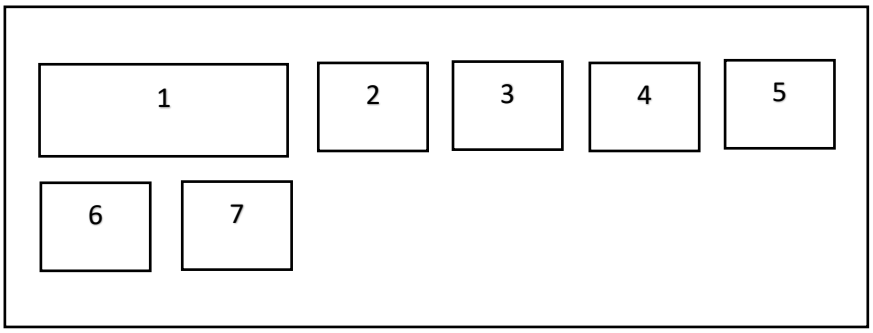
The column in each row is defined inside the apostrophes, separated by space.

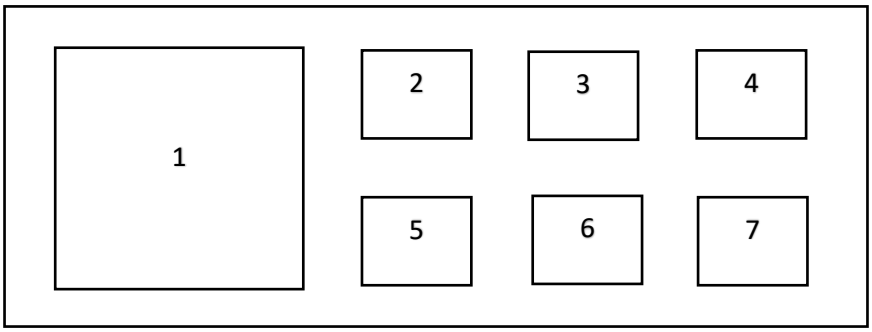
Note 3

A period sign represents a grid item with no name.

Example 1:

```
.Item1
    {
      grid-area: myarea;
     }
.container
     {
      display: grid;
      grid-template-areas: `myarea myarea ……`;
     }
```

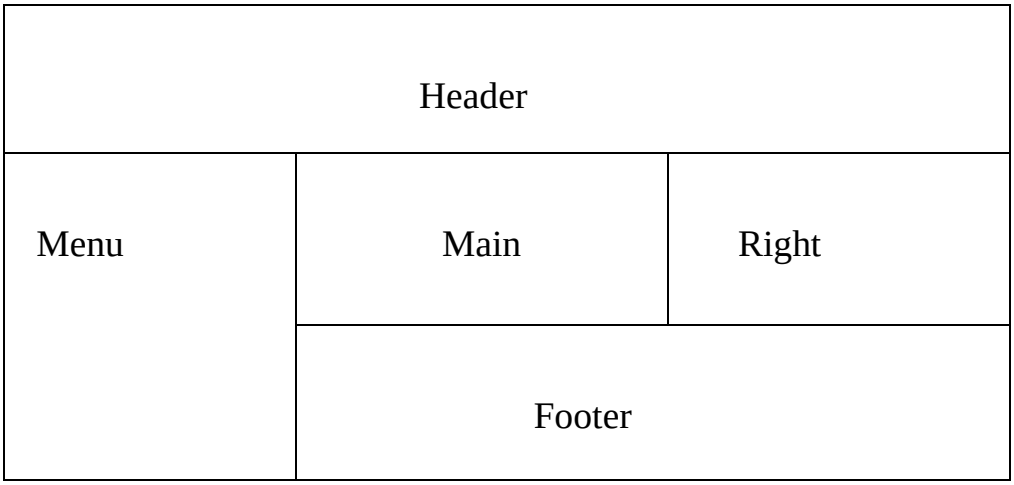




Example 2:

```
.container
    {
      display: grid;
      grid-template-areas: `myarea myarea ……`
                           `myarea myarea ……`;
      }
```

Example:




```
<style>
 .Item1
   {
      grid-area: header;
    }
  .Item2
   {
      grid-area: menu;
    }
 .Item3
   {
      grid-area: main;
    }
 .Item4
   {
      grid-area: Right;
    }
  .Item5
```

```
{
   grid-area: Footer;
 }
.gridcontainer
        {
           display: grid;
           grid-template-areas: `header header header header
                                 header header` `menu main main
                                 Right Right` `menu footer footer
                                  footer footer footer`;
        }
```

Order Of the Item

Example:

```
.Item1
   {
      grid-row-start: 2;
      grid-row-end: 3;
      grid-column-start: 2;
      grid-column-end: 3;
    }
```

It`s equal to: .Item1{ grid-area: 2 / 2 / 3 / 3 }

Example: .Item1{ grid-area: 1 / 3 / 2 / 4; }

.Item2{ grid-area: 2 / 3 / 3 / 4; }

.Item3{ grid-area: 1 / 1 / 2 / 2; }

.Item4{ grid-area: 1 / 2 / 2 / 3; }

.Item5{ grid-area: 2 / 1 / 3 / 2; }

.Item6{ grid-area: 2 / 2 / 3 / 4; }

3	4	1
5	6	2

Image Gallery Using Grid Layout

<table>
<tr><td colspan="2" rowspan="2">Image1</td><td rowspan="2">Image2</td><td colspan="2">Image3</td></tr>
<tr><td>6</td><td rowspan="2">7</td></tr>
<tr><td rowspan="2">4</td><td rowspan="3" colspan="2">5</td><td>10</td></tr>
<tr><td rowspan="2">17</td><td rowspan="3" colspan="2">11</td></tr>
<tr><td rowspan="4">8</td><td colspan="2">9</td></tr>
<tr><td rowspan="3">12</td><td rowspan="3">13</td><td rowspan="3">14</td><td>15</td></tr>
<tr><td rowspan="2">16</td></tr>
</table>